



FFFFFFFF	AAAAAA	LL	AAAAAA	CCCCCCCC	TTTTTTTT	MM	MM	SSSSSSSS	GGGGGGGG
FFFFFFFF	AAAAAA	LL	AAAAAA	CCCCCCCC	TTTTTTTT	MM	MM	SSSSSSSS	GGGGGGGG
FF	AA	AA	AA	CC	TT	MMM	MMM	SS	GG
FF	AA	AA	AA	CC	TT	MMM	MMM	SS	GG
FF	AA	AA	AA	CC	TT	MM	MM	SS	GG
FF	AA	AA	AA	CC	TT	MM	MM	SS	GG
FFFFFFFF	AA	AA	AA	CC	TT	MM	MM	SSSSSS	GG
FFFFFFFF	AA	AA	AA	CC	TT	MM	MM	SSSSSS	GG
FF	AAAAAAAA	LL	AAAAAAAA	CC	TT	MM	MM	SS	GG
FF	AAAAAAAA	LL	AAAAAAAA	CC	TT	MM	MM	SS	GG
FF	AA	AA	AA	CC	TT	MM	MM	SS	GG
FF	AA	AA	AA	CC	TT	MM	MM	SS	GG
FF	AA	AA	LLLLLLLLLL	CCCCCCCC	TT	MM	MM	SSSSSSSS	GGGGGG
FF	AA	AA	LLLLLLLLLL	CCCCCCCC	TT	MM	MM	SSSSSSSS	GGGGGG

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```



(2)	66	DECLARATIONS
(3)	113	ACTION ROUTINES
(4)	164	FALS\$DECODE_CNF
(5)	248	FALS\$DECODE_ATT
(6)	328	FALS\$DECODE_ACC
(7)	405	FALS\$DECODE_CTL
(8)	616	FALS\$DECODE_CON
(9)	632	FALS\$DECODE_CMP
(10)	659	FALS\$DECODE_KEY
(11)	741	FALS\$DECODE_ALL
(12)	802	FALS\$DECODE_TIM
(13)	848	FALS\$DECODE_PRO
(14)	919	FALS\$DECODE_NAM
(15)	941	SUPPORT ROUTINES
(15)	944	MAP_FOP_FIELD
(16)	988	MAP_ROP_FIELD
(17)	1024	STATE EXIT ROUTINES

```
0000 1 .TITLE FALACTMSG - STATE TABLE ACTION ROUTINES
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 Facility: FAL (DECnet File Access Listener)
0000 31
0000 32 Abstract:
0000 33
0000 34 This module contains action routines called by the state table manager.
0000 35
0000 36 Environment: VAX/VMS, user mode
0000 37
0000 38 Author: James A. Krycka, Creation Date: 16-JUN-1977
0000 39
0000 40 Modified By:
0000 41
0000 42 V03-007 JAK0136 J A Krycka 07-MAR-1984
0000 43 Support FAL logging options that deal with fields in the DAP
0000 44 Configuration message sent to partner.
0000 45
0000 46 V03-006 JAK0118 J A Krycka 12-JUL-1983
0000 47 Fix bug in processing the DAP KEY field.
0000 48
0000 49 V03-005 KRM0104 K Malik 10-May-1983
0000 50 Update symbols to match Dap V7.0 spec.
0000 51
0000 52 V03-004 JAK0104 J A Krycka 29-APR-1983
0000 53 Make minor enhancements to FAL logging display.
0000 54
0000 55 V03-003 KRM0083 K Malik 23-Mar-1983
0000 56 Add support for STMLF and STMCR formats.
0000 57
```



0000 58 :  
0000 59 :  
0000 60 :  
0000 61 :  
0000 62 :  
0000 63 :  
0000 64 :--

V03-002 KRM0074 K Malik 23-Nov-1982  
Added FAL\$DECODE\_NAM routine (to support \$RENAME function).  
V03-001 JAK0101 J A Krycka 09-OCT-1982  
Fix bug in converting DAP OWNER value into binary format.

```
0000 66 .SBTTL DECLARATIONS
0000 67
0000 68 :
0000 69 : Include Files:
0000 70 :
0000 71
0000 72 $DAPPLGDEF ; Define DAP prologue symbols
0000 73 $DAPHDRDEF ; Define DAP message header
0000 74 $DAPSSPDEF ; Define DAP system specific field
0000 75 $DAPCNFDEF ; Define DAP Configuration message
0000 76 $DAPATTDEF ; Define DAP Attributes message
0000 77 $DAPACCDEF ; Define DAP Access message
0000 78 $DAPCTLDEF ; Define DAP Control message
0000 79 $DAPCONDEF ; Define DAP Continue Transfer message
0000 80 $DAPCMPDEF ; Define DAP Access Complete message
0000 81 $DAPKEYDEF ; Define DAP Key Definition message
0000 82 $DAPALLDEF ; Define DAP Allocation message
0000 83 $DAPTIMDEF ; Define DAP Date and Time message
0000 84 $DAPPRODEF ; Define DAP Protection message
0000 85 $DAPNAMDEF ; Define DAP Name message
0000 86 $DEVDEF ; Define Device Characteristics symbols
0000 87 $FABDEF ; Define File Access Block symbols
0000 88 $FALWRKDEF ; Define FAL Work Area symbols
0000 89 $RABDEF ; Define Record Access Block sym**
0000 90 $XABDEF ; Define symbols common to all XABs
0000 91 $XABALLDEF ; Define Allocation XAB symbols
0000 92 $XABDATDEF ; Define Date and Time XAB symbols
0000 93 $XABKEYDEF ; Define Key Definition XAB symbols
0000 94 $XABPRODEF ; Define Protection XAB symbols
0000 95 : $XABRDTDEF ; Define Revision Date and Time symbols
0000 96
0000 97 :
0000 98 : Macros:
0000 99 :
0000 100 : None
0000 101 :
0000 102 : Equated Symbols:
0000 103 :
0000 104 :
0000 105
0000 106 ASSUME DAP$Q_DCODE FLG EQ 0
0000 107 ASSUME FAL$Q_FLG EQ 0
0000 108
0000 109 :
0000 110 : Own Storage:
0000 111 :
```



```
0000 113 .SBTTL ACTION ROUTINES
00000000 114 .PSECT FAL$CODE NOSHR,EXE,RD,NOWRT,BYTE
0000 115
0000 116 :++
0000 117 : Functional Description:
0000 118 :
0000 119 : This module contains action routines invoked by the state table
0000 120 : manager (FAL$STATE).
0000 121 :
0000 122 : The input parameters and completion codes listed below are applicable
0000 123 : for all of these action routines. Note that an action routine may use
0000 124 : R0-R7 and AP without restoring them on exit. R0 on exit, however, must
0000 125 : represent a status code to indicate success/failure of the routine or
0000 126 : a true/false condition, as appropriate. This status code is used by
0000 127 : the state table manager to advance to the next state.
0000 128 :
0000 129 : Calling Sequence:
0000 130 :
0000 131 : BSBW FAL$name
0000 132 :
0000 133 : Input Parameters:
0000 134 :
0000 135 : R8 Address of FAL work area
0000 136 : R9 Address of DAP control block
0000 137 : R10 Address of FAB
0000 138 : R11 Address of RAB
0000 139 :
0000 140 : Implicit Inputs:
0000 141 :
0000 142 : None
0000 143 :
0000 144 : Output Parameters:
0000 145 :
0000 146 : R0 Status code
0000 147 : R1-R7 Destroyed
0000 148 : AP Destroyed
0000 149 :
0000 150 : Implicit Outputs:
0000 151 :
0000 152 : None
0000 153 :
0000 154 : Completion Codes:
0000 155 :
0000 156 : R0 1 = success; 0 = failure
0000 157 :
0000 158 : Side Effects:
0000 159 :
0000 160 : None
0000 161 :
0000 162 :--
```

```
0000 164 .SBTTL FALS$DECODE_CNF
0000 165
0000 166 :++
0000 167 : Process the Configuration message which has been received and validated.
0000 168 : Return a Configuration message to partner and determine the DAP buffer size
0000 169 : to use which is the smaller of partner's buffer size and FAL's buffer size.
0000 170 :--
0000 171
0000 172 FALS$DECODE_CNF:: : Entry point
0000 173 $SETBIT #FALS$V_CNF_MSG,(R8) : Denote Configuration message received
0004 174 $CLRBIT #FALS$V_ATT_MSG,(R8) : and discard any previous Attributes
0008 175 : message
57 18 A8 3C 0008 176 MOVZWL FALS$W_QIOBUFSIZ(R8),R7 : Get FAL's buffer size (i.e., largest
000C 177 : I/O buffer size supported by process)
68 39 E1 000C 178 BBC #FALS$V_USE_DBS,(R8),- : Branch to use calculated buffer size
05 000F 179 SEND CNF
57 00A0 C8 3C 0010 180 MOVZWL FALS$W_USE_DBS(R8),R7 : Override with user specified value
0015 181
0015 182 :+
0015 183 : Build and send Configuration message to partner.
0015 184 :--
0015 185
0015 186 SEND_CNF:
0015 187 $SETBIT #FALS$V_LAST_MSG,(R8) : Declare this last message to block
50 01 D0 0019 188 MOVL #DAP$K_CNF_MSG,R0 : Get message type value
FFE1' 30 001C 189 BSBW FALS$BUILT_HEAD : Construct message header
83 57 B0 001F 190 MOVW R7,(R3)+ : Store BUFSIZ field
83 07 90 0022 191 MOVW #DAP$K_VAXVMS,(R3)+ : Store OSTYPE field
83 03 90 0025 192 MOVW #DAP$K_RMS32,(R3)+ : Store FILESYS field
06 68 3A E1 0028 193 BBC #FALS$V_USE_SYS,(R8),2$ : Branch to use standard values
FE A3 00A2 C8 B0 002C 194 MOVW FALS$W_USE_SYS(R8),-2(R3) : Override with user specified values
83 07 90 0032 195 2$: MOVW #DAP$K_VERNUM_V,(R3)+ : Store VERNUM field
83 00 90 0035 196 MOVW #DAP$K_ECONUM_V,(R3)+ : Store ECONUM field
83 C0 90 0038 197 MOVW #DAP$K_USRNUM_V,(R3)+ : Store USRNUM field
83 04 90 003B 198 MOVW #DAP$K_DECVER_V,(R3)+ : Store DECVER field
06 68 3B E1 003E 199 BBC #FALS$V_USE_VER,(R8),4$ : Branch to use standard values
FC A3 00A4 C8 D0 0042 200 MOVL FALS$L_USE_VER(R8),-4(R3) : Override with user specified values
83 00 90 0048 201 4$: MOVW #DAP$K_USRVER_V,(R3)+ : Store USRVER field
004B 202
004B 203 :
004B 204 : Construct the system capabilities field.
004B 205 : Also, check the debugging options to disable message blocking and DAP level
004B 206 : CRC checking (after any user specified system capabilities bitmasks, if any,
004B 207 : have been applied).
004B 208 :
004B 209
51 EFF67DF7 8F D0 004B 210 MOVL #DAP$K_SYSCAP1_V,R1 : Get VAX supported capabilities
52 00001962 8F D0 0052 211 MOVL #DAP$K_SYSCAP2_V,R2 : quadword bitmask
0059 212 : ----- process debugging options -----
05 68 3C E1 0059 213 BBC #FALS$V_USE_SC1,(R8),6$ : Branch to use standard values
51 00A8 C8 D0 005D 214 MOVL FALS$L_USE_SC1(R8),R1 : Override with user specified values
05 68 3D E1 0062 215 6$: BBC #FALS$V_USE_SC2,(R8),8$ : Branch to use standard values
52 00AC C8 D0 0066 216 MOVL FALS$L_USE_SC2(R8),R2 : Override with user specified values
0F 68 31 E1 006B 217 8$: BBC #FALS$V_DIS_MBK,(R8),10$ : Is DAP message blocking disabled?
51 00140000 8F CA 006F 218 BICL2 #<<1aDAP$V_MSGBLK>!-- : Yes, clear message blocking bits in
0076 219 : system capabilities bitmask for
0076 220 : Configuration message to transmit
```



```
28 A9 00140000 8F CA 0076 221 BICL2 #<<1@DAP$V_MSGBLK>!-- ; Also, clear message blocking bits in
                                007E 222 ; <1@DAP$V_BIGBLK>!-- ; system capabilities bitmask
                                007E 223 ; 0>,DAP$Q_SYSCAP(R9) ; received from partner
                                09 68 30 E1 007E 224 10$: BBC #FALS$V_DIS_CRC,(R8),20$ ; Is file level CRC checksum disabled?
                                0082 225 $CLRBIT #DAP$V-DAPCRC,R1 ; Yes, clear bits in both XMT and RCV
                                0086 226 $CLRBIT #DAP$V-DAPCRC,- ; system capabilities fields
                                0086 227 DAP$Q_SYSCAP(R9) ;
                                008B 228 ; ----- finish debugging options -----
                                FF72' 30 008B 229 20$: BSBW FALS$CVT_BN8_EXT ; Store SYSCAP as an extensible field
                                FF6F' 30 008E 230 BSBW FALS$BUICD_TAIL ; Finish building message
                                FF6C' 30 0091 231 BSBW FALS$TRANSMIT ; Send Configuration message
                                0094 232 ;
                                0094 233 ;+
                                0094 234 ; Determine the 'agreed upon' DAP buffer size to use and save this value.
                                0094 235 ; It is the smaller of partner's buffer size and FAL's maximum buffer size.
                                0094 236 ;-
                                0094 237
                                40 A9 B0 0094 238 MOVW DAP$W_BUFSIZ(R9),- ; Assume we'll use partner's
                                1A A8 0097 239 FALS$W_DAPBUFSIZ(R8) ; buffer size
                                06 13 0099 240 BEQL 30$ ; Branch if partner has unlimited space
                                57 40 A9 B1 009B 241 CMPW DAP$W_BUFSIZ(R9),R7 ; Compare partner's buffer size with
                                009F 242 ; our buffer size
                                04 1B 009F 243 BLEQU 40$ ; Branch if partner has less capacity
                                1A A8 57 B0 00A1 244 30$: MOVW R7,FALS$W_DAPBUFSIZ(R8) ; We guessed wrong, so we'll use
                                00A5 245 ; our buffer size
                                04E4 31 00A5 246 40$: BRW EXIT_SUCCESS ; Exit state with success
```

```
00A8 248 .SBTTL FAL$DECODE_ATT
00A8 249
00A8 250 :++
00A8 251 : Process the Attributes message which has been received and validated.
00A8 252 : Update the FAB and FHCXAB with information from this message.
00A8 253 :--
00A8 254
00A8 255 FAL$DECODE_ATT:: ; Entry point
00A8 256
00A8 257 $SETBIT #FAL$V_ATT_MSG,(R8) ; Denote Attributes message received
00AC 258
00AC 259 :
00AC 260 : Save the DAP DATATYPE field for use later.
00AC 261 :
00AC 262
01F4 C8 44 A9 90 00AC 263 MOVB DAP$B_DATATYPE(R9),FAL$B_DATATYPE(R8)
00B2 264
00B2 265 :
00B2 266 : Process the DAP ORG, RFM and RAT fields.
00B2 267 :
00B2 268
00B2 269 ASSUME DAP$K_SEQ EQ FAB$C_SEQ
00B2 270 ASSUME DAP$K_REL EQ FAB$C_REL
00B2 271 ASSUME DAP$K_IDX EQ FAB$C_IDX
00B2 272
1D AA 45 A9 90 00B2 273 MOVB DAP$B_ORG(R9),FAB$B_ORG(R10)
00B7 274
00B7 275 ASSUME DAP$K_UDF EQ FAB$C_UDF
00B7 276 ASSUME DAP$K_FIX EQ FAB$C_FIX
00B7 277 ASSUME DAP$K_VAR EQ FAB$C_VAR
00B7 278 ASSUME DAP$K_VFC EQ FAB$C_VFC
00B7 279 ASSUME DAP$K_STM EQ FAB$C_STM
00B7 280 ASSUME DAP$K_STMLF EQ FAB$C_STMLF
00B7 281 ASSUME DAP$K_STMCR EQ FAB$C_STMCR
00B7 282
1F AA 46 A9 90 00B7 283 MOVB DAP$B_RFM(R9),FAB$B_RFM(R10)
00BC 284
00BC 285 ASSUME DAP$V_FTN EQ FAB$V_FTN
00BC 286 ASSUME DAP$V_CR EQ FAB$V_CR
00BC 287 ASSUME DAP$V_PRN EQ FAB$V_PRN
00BC 288 ASSUME DAP$V_BLK EQ FAB$V_BLK
00BC 289
1E AA 47 A9 90 00BC 290 MOVB DAP$B_RAT(R9),FAB$B_RAT(R10)
00C1 291 BICB2 #DAP$M_EMBEDDED,- ; Ignore this bit
00C3 292 FAB$B_RAT(R10)
00C5 293 BBS #DAP$V_VAXVMS,(R9),10$ ; Branch if partner is VAX/VMS
0A 69 34 E0 00C9 294 CMPB DAP$B_RFM(R9),#DAP$K_STM; Branch if not stream format
04 46 A9 91 00CD 295 BNEQ 10$
00CF 296 MOVB #FAB$M_CR,FAB$B_RAT(R10); If it is, declare cc to be implied
1E AA 02 90 00D3 297
00D3 298 :
00D3 299 : Process the DAP BLS, MRS, ALQ, BKS, FSZ, MRN, and DEQ fields.
00D3 300 :
00D3 301
3C AA 48 A9 B0 00D3 302 10$: MOVW DAP$W_BLS(R9),FAB$W_BLS(R10)
36 AA 4A A9 B0 00D8 303 MOVW DAP$W_MRS(R9),FAB$W_MRS(R10)
10 AA 4C A9 D0 00DD 304 MOVL DAP$L_ALQ1(R9),FAB$L_ALQ(R10)
```



```
3E AA 50 A9 90 00E2 305      MOVB  DAP$B_BKS(R9),FAB$B_BKS(R10)
3F AA 51 A9 90 00E7 306      MOVB  DAP$B_FSZ(R9),FAB$B_FSZ(R10)
38 AA 58 A9 D0 00EC 307      MOVL  DAP$L_MRN(R9),FAB$L_MRN(R10)
14 AA 54 A9 B0 00F1 308      MOVW  DAP$W_DEQ1(R9),FAB$W_DEQ(R10)
      00F6 309
      00F6 310
      00F6 311 : Process the DAP FOP field after saving it for use later.
      00F6 312 :
      00F6 313
51 64 A9 D0 00F6 314      MOVL  DAP$L_FOP1(R9),R1      ; Get DAP FOP bits and
01F8 C8 51 D0 00FA 315      MOVL  R1,FAL$L_FOP(R8)      ; save field for use later
      0342 30 00FF 316      BSBW  MAP_FOP_FIELD      ; Update FOP in FAB
      0102 317
      0102 318
      0102 319 : Process the DAP LRL field.
      0102 320 : This is the only FHCXAB field that is input to RMS, and then only for the
      0102 321 : $CREATE function where the record format is variable or VFC.
      0102 322 :
      0102 323
      70 A9 B0 0102 324      MOVW  DAP$W_LRL(R9),-      ; Copy value to FHCXAB
02F4 C8 0105 325      FAL$L_FHCXAB+XAB$W_LRL(R8)
      0481 31 0108 326      BRW   EXIT_SUCCESS      ; Exit state with success
```

```
010B 328 .SBTTL FAL$DECODE_ACC
010B 329
010B 330 :++
010B 331 : Process the Access message which has been received and validated.
010B 332 : Update the FAB with information from this message.
010B 333 :--
010B 334
010B 335 FAL$DECODE_ACC:: ; Entry point
010B 336
010B 337 :
010B 338 : Save the DAP ACCFUNC, ACCOPT, and DISPLAY fields for use later.
010B 339 :
010B 340
01F6 C8 40 A9 90 010B 341 MOVB DAP$B_ACCFUNC(R9),FAL$B_ACCFUNC(R8)
01F5 C8 41 A9 90 0111 342 MOVB DAP$B_ACCOPT(R9),FAL$B_ACCOPT(R8)
70 AB 4C A9 B0 0117 343 MOVW DAP$W_DISPLAY1(R9),FAL$W_DISPLAY(R8)
011C 344
011C 345 :
011C 346 : Process the DAP file specification field.
011C 347 :
011C 348
44 A9 90 011C 349 MOVB DAP$Q_FILESPEC(R9),- ; Store size of filespec string
34 AA 011F 350 FAB$B_FNS(R10) ; in FAB
44 A9 28 0121 351 MOV C3 DAP$Q_FILESPEC(R9),- ; Copy filespec string to buffer
48 B9 0124 352 @DAP$Q_FILESPEC+4(R9),- ;
2C BA 0126 353 @FAB$B_FNA(R10) ;
51 40 A9 9A 0128 354 MOVZBL DAP$B_ACCFUNC(R9),R1 ; Get access function code
52 44 A9 7E 012C 355 MOVAQ DAP$Q_FILESPEC(R9),R2 ; Get address of filename descriptor
FECD' 30 0130 356 BSBW FAL$LOG_REQNAM ; Log requested name in print file
0133 357
0133 358 :
0133 359 : Process the DAP FAC field.
0133 360 :
0133 361
0133 362 ASSUME DAP$V_PUT EQ FAB$V_PUT
0133 363 ASSUME DAP$V_GET EQ FAB$V_GET
0133 364 ASSUME DAP$V_DEL EQ FAB$V_DEL
0133 365 ASSUME DAP$V_UPD EQ FAB$V_UPD
0133 366 ASSUME DAP$V_TRN EQ FAB$V_TRN
0133 367 ASSUME DAP$V_BIO EQ FAB$V_BIO
0133 368 ASSUME DAP$V_BRO EQ FAB$V_BRO
0133 369 ASSUME DAP$V_APP EQ FAB$V_EXE ; Map APP to PUT
0133 370
16 AA 42 A9 90 0133 371 MOVB DAP$B_FAC(R9),FAB$B_FAC(R10)
05 16 AA 07 E5 0138 372 BBCC #FAB$V_EXE,FAB$B_FAC(R10),10$
00 16 AA 00 E2 013D 373 BBSS #FAB$V_PUT,FAB$B_FAC(R10),10$
0142 374
0142 375 :
0142 376 : Process the DAP SHR field.
0142 377 :
0142 378
51 43 52 D4 0142 379 10$: CLRL R2 ; Clear RMS SHR bits
A9 9A 0144 380 MOVZBL DAP$B_SHR(R9),R1 ; Get DAP SHR bits
30 13 0148 381 BEQL 20$ ; Branch if no bits to map
014A 382 $MAPBIT DAP$V_SHRPUT,FAB$V_SHRPUT; Map SHRPUT bit
0152 383 $MAPBIT DAP$V_SHRGET,FAB$V_SHRGET; Map SHRGET bit
015A 384 $MAPBIT DAP$V_SHRDEL,FAB$V_SHRDEL; Map SHRDEL bit
```



```

      0162 385 $MAPBIT DAP$V_SHRUPD,FAB$V_SHRUPD; Map SHRUPD bit
      016A 386 $MAPBIT DAP$V_UPI,FAB$V_UPI; Map UPI bit
      0172 387 $MAPBIT DAP$V_NIL,FAB$V_NIL; Map NIL bit
17 AA 52 90 017A 388 20$: MOVB R2,FAB$B_SHR(R10); Update SHR field in FAB
      017E 389
      017E 390
      017E 391 :: Use the ACCFUNC field value as the next state table value.
      017E 392 ::
      017E 393
      40 A9 90 017E 394 MOVB DAP$B_ACCFUNC(R9),-; Store new state transition value
      10 A8 90 0181 395 FAL$B_VALUE(R8);
      0A 0084 C9 E1 0183 396 BBC #DAP$V_LOAD,-; Branch if no system specific
      0185 397 DAP$L_SSP_FLG(R9),30$; function modifier found
      0189 398 $SETBIT #FAB$V_SQ0,FAB$B_FOP(R10);
      018E 399; Force sequential file transfer
      018E 400; mode for efficiency
      FF 8F 90 018E 401 MOVB #DAP$K_LOAD,-; Make new state transition value
      10 A8 90 0191 402 FAL$B_VALUE(R8); the load image function
      03F6 31 0193 403 30$: BRW EXIT_SUCCESS; Exit state with success
```

```
0196 405 .SBTTL FALS$DECODE_CTL
0196 406
0196 407 :++
0196 408 : Process the Control message which has been received and validated.
0196 409 : Update the RAB with information from this message.
0196 410 :--
0196 411
0196 412 FALS$DECODE_CTL:: ; Entry point
0196 413
0196 414 :+
0196 415 : Save the DAP DISPLAY field for use later if we're not in a wildcard context.
0196 416 : In wildcard file retrieval, for example, the DAP Access message is sent only
0196 417 : once, thus FALS$W_DISPLAY must reflect the DISPLAY value from the Access
0196 418 : message on subsequent file opens. Since the Control message functions of
0196 419 : DISPLAY and EXTEND are not valid in a wildcard context (which require
0196 420 : FALS$W_DISPLAY to be updated), this special check is an acceptable solution
0196 421 : to a wildcard retrieval problem.
0196 422 :--
0196 423
68 0A E0 0196 424 BBS #FALS$W_WILD,(R8),- ; Branch if wildcard operation
05 05 0199 425 RAC_FIELD ;
54 A9 B0 019A 426 MOVW DAP$W_DISPLAY2(R9),- ; Save display message bitmask in
70 A8 019D 427 FALS$W_DISPLAY(R8) ; FAL work area
019F 428
019F 429 :+
019F 430 : Process the DAP RAC field.
019F 431 : In addition to normal RMS-32 RAC information, this field specifies whether
019F 432 : the access is to be in:
019F 433 : (1) file transfer mode or record transfer mode
019F 434 : (2) block I/O mode or record I/O mode
019F 435 :
019F 436 : Note: If the RAC field is not present in the Control message, then the default
019F 437 : action is to use the previous value.
019F 438 :--
019F 439
019F 440 ASSUME DAP$K_SEQ_ACC EQ 0
019F 441 ASSUME DAP$K_KEY_ACC EQ 1
019F 442 ASSUME DAP$K_RFA_ACC EQ 2
019F 443 ASSUME DAP$K_SEQ_FILE EQ 3
019F 444 ASSUME DAP$K_BLK_VBN EQ 4
019F 445 ASSUME DAP$K_BLK_FILE EQ 5
019F 446
019F 447 RAC_FIELD:
019F 448 BBS #DAP$V_RAC,- ; Process record access field
06 44 A9 E0 01A1 449 DAP$W_CTLMENU(R9),10$ ; Branch if RAC field was explicitly
01F7 C8 90 01A4 450 MOVW FALS$B_RAC(R8),- ; specified
46 A9 90 01A8 451 DAP$B_RAC(R9) ; If not, use previous value saved in
46 A9 90 01AA 452 10$: MOVW DAP$B_RAC(R9),- ; FAL work area
01F7 C8 01AD 453 FALS$B_RAC(R8) ; Save currently specified value as
01B0 454 $CASEB SELECTOR=DAP$B_RAC(R9)- ; previous value for next-time-thru
01B0 455 DISPL=<- ; Dispatch on DAP record access mode:
01B0 456 20$- ; Sequential record access
01B0 457 20$- ; Random access by key value
01B0 458 20$- ; Random access by RFA
01B0 459 30$- ; Sequential file transfer
01B0 460 40$- ; Block I/O access by VBN
01B0 461 50$- ; Block I/O sequential file transfer
```



```
01B0 462 > ;
01C1 463 ;
01C1 464 : Update the RAC field of the RAB unless block I/O mode is specified.
01C1 465 : (RMS-32 ignores the RAC field on block I/O operations.)
01C1 466 ;
01C1 467 : Also update the file transfer mode and block I/O flags as appropriate
01C1 468 : for the access mode invoked.
01C1 469 ;
01C1 470 ;
01C1 471 ASSUME DAP$K_SEQ_ACC EQ RAB$C_SEQ
01C1 472 ASSUME DAP$K_KEY_ACC EQ RAB$C_KEY
01C1 473 ASSUME DAP$K_RFA_ACC EQ RAB$C_RFA
01C1 474 ;
46 A9 90 01C1 475 20$: MOV B DAP$B_RAC(R9),- ; Store record access mode in RAB
1E AB 01C4 476 RAB$B_RAC(R11) ; (either SEQ, KEY, or RFA)
08 11 01C6 477 $CLRBIT #FALS$V_FTM,(R8) ; Say record transfer mode
1E AB 00 90 01CA 478 BRB 35$ ;
01CC 479 30$: MOV B #RAB$C_SEQ,RAB$B_RAC(R11) ; Set record access mode to SEQ in RAB
01D0 480 $SETBIT #FALS$V_FTM,(R8) ; Say file transfer mode
01D4 481 35$: $CLRBIT #FALS$V_BLK_IO,(R8) ; Say record I/O mode
0E 11 01D8 482 BRB ROP_FIELD ;
01DA 483 40$: $CLRBIT #FALS$V_FTM,(R8) ; Say record transfer mode
04 11 01DE 484 BRB 55$ ;
01E0 485 50$: $SETBIT #FALS$V_FTM,(R8) ; Say file transfer mode
01E4 486 55$: $SETBIT #FALS$V_BLK_IO,(R8) ; Say block I/O mode
01E8 487 ;
01E8 488 :+
01E8 489 : Process the DAP ROP field.
01E8 490 ;
01E8 491 : Note: If the ROP field is not present in the Control message, then the default
01E8 492 : action is to use the previous value.
01E8 493 :-
01E8 494 ;
01E8 495 ROP_FIELD: ; Process record options field
03 E1 01E8 496 BBC #DAP$V_ROP,- ; Branch if ROP field was not explicitly
44 A9 01EA 497 DAP$W_TLMENU(R9),- ; specified making previous ROP value
07 01EC 498 KRF_FIELD ; the current value
51 50 A9 D0 01ED 499 MOVL DAP$L_ROP(R9),R1 ; Get DAP ROP bits
02F7 30 01F1 500 BSBW MAP_ROP_FIELD ; Update ROP options in RAB
01F4 501 ;
01F4 502 :+
01F4 503 : Process the DAP KRF field.
01F4 504 : This field is applicable only for indexed files.
01F4 505 ;
01F4 506 : Note: If the KRF field is not present in the Control message, then the default
01F4 507 : action is to use the previous value.
01F4 508 :-
01F4 509 ;
01F4 510 KRF_FIELD: ; Process key of reference field
02 E1 01F4 511 BBC #DAP$V_KRF,- ; Branch if KRF field was not explicitly
44 A9 01F6 512 DAP$W_TLMENU(R9),- ; specified making previous KRF value
05 01F8 513 KEY_FIELD ; the current value
47 A9 90 01F9 514 MOV B DAP$B_KRF(R9),- ; Update key of reference value in RAB
35 AB 01FC 515 RAB$B_KRF(R11) ; (meaningful only for indexed files)
01FE 516 ;
01FE 517 :+
01FE 518 : Process the DAP KEY field.
```

```
01FE 519 : Its format and content are context dependent:
01FE 520 : (1) for block I/O access, it contains the virtual block number for
01FE 521 : $READ/$WRITE, or the number of blocks for $SPACE.
01FE 522 : (2a) for sequential record access without the key limit option in force,
01FE 523 : this field is ignored because RMS will use its internally stored
01FE 524 : next-record-pointer to locate the record.
01FE 525 : (2b) for sequential record access of an indexed file with the key limit
01FE 526 : option set (i.e., ROP = LIM), it contains the key value string.
01FE 527 : (3a) for random access by key value for relative (or fixed length
01FE 528 : sequential) files, it contains the relative record number.
01FE 529 : (3b) for random access by key value for indexed files, it contains the
01FE 530 : key value string.
01FE 531 : (4) for random access by record file address, it contains the RFA value.
01FE 532 :-
01FE 533
01FE 534 KEY_FIELD: ; Process the key field
50 48 A9 7D 01FE 535 MOVQ DAP$Q_KEY(R9),R0 ; <R0,R1> => descriptor of key field
47 68 09 E0 0202 536 BBS #FALS$V_BLK_IO,(R8),50$ ; Branch if block I/O access
0206 537
0206 538 ASSUME RAB$C_SEQ EQ 0
0206 539 ASSUME RAB$C_KEY EQ 1
0206 540 ASSUME RAB$C_RFA EQ 2
0206 541
0206 542 $CASEB SELECTOR=RAB$B_RAC(R11)-; Dispatch on RMS record access mode:
0206 543 BASE=#RAB$C_SEQ-
0206 544 DISPL=<-
0206 545 10$- ; Sequential record access
0206 546 20$- ; Random access by key value
0206 547 40$- ; Random access by RFA
0206 548 >
1C 44 01 E0 0211 549 10$: BBS #DAP$V_KEY,- ; Update key value only if KEY field
4D 11 0213 550 DAP$W_CTLMENU(R9),30$ ; was explicitly specified
20 1D AA 91 0216 551 BRB 90$ ; All done with key field
14 13 0218 552 20$: CMPB FAB$B_ORG(R10),#FAB$C_IDX; Branch if indexed file
021C 553 BEQL 30$ ; Fall thru if sequential or relative
021E 554
021E 555 :
021E 556 : Key field contains a relative record number (RRN).
021E 557 : RMS requires that the RRN be a 4-byte unsigned integer value.
021E 558 :
021E 559
34 AB 04 90 021E 560 MOVB #4,RAB$B_KSZ(R11) ; Store size and address of buffer
01FC C8 DE 0222 561 MOVAL FALS$_NUMBER(R8),- ; that will hold RRN value
30 AB 0226 562 RAB$B_KBF(R11) ; in KSZ/KBF fields of RAB
04 00 61 50 2C 0228 563 MOVCS R0,(RT),#0,#4,- ; Copy RRN value as a longword to
01FC C8 022D 564 FALS$_NUMBER(R8) ; buffer in FAL work area
22 11 0230 565 BRB 60$ ; Join common code
0232 566
0232 567 :
0232 568 : Key field contains a key string.
0232 569 :
0232 570
34 AB 50 90 0232 571 30$: MOVB R0,RAB$B_KSZ(R11) ; Store size and address of buffer
0700 C8 DE 0236 572 MOVAL FALS$_KEYBUF(R8),- ; that will hold key string value
30 AB 023A 573 RAB$B_KBF(R11) ; in KSZ/KBF fields of RAB
0700 C8 61 50 28 023C 574 MOVCS R0,(RT),FALS$_KEYBUF(R8); Copy string to buffer in FAL work area
21 11 0242 575 BRB 90$ ; All done with key field
```



```
0244 576
0244 577 :
0244 578 : Key field contains a record file address (RFA).
0244 579 : RMS requires that the RFA be a 6-byte unsigned integer value.
0244 580 :
0244 581 :
06 00 61 50 2C 0244 582 40$:  MOVCS  R0,(R1),#0,#6,-      ; Store RFA value directly in RFA field
      10 AB      0249 583      RAB$W_RFA(R11)      ; of RAB (zero extended to 6-bytes)
      07 11 024B 584      BRB      60$            ; Join common code
      024D 585 :
      024D 586 :
      024D 587 : Key field contains virtual block number (VBN).
      024D 588 : RMS requires that the VBN be a 4-byte unsigned integer value.
      024D 589 :
      024D 590 :
04 00 61 50 2C 024D 591 50$:  MOVCS  R0,(R1),#0,#4,-      ; Store VBN value directly in BKT field
      38 AB      0252 592      RAB$L_BKT(R11)      ; of RAB (zero extended to longword)
      0254 593 :
      0254 594 :
      0254 595 : Common code to verify that the length of the string in the DAP KEY field
      0254 596 : does not exceed the size of the buffer used to store the string.
      0254 597 :
      0254 598 :
      0F 1B 0254 599 60$:  BLEQU  90$            ; Done if all SRC bytes are copied
      0256 600      : (i.e., SRC size LEQU DST size)
      81 95 0256 601 70$:  TSTB   (R1)+          ; Error if any unmoved bytes are
      05 12 0258 602      BNEQ   80$            ; non-zero
      F9 50 F5 025A 603      SOBGTR R0,70$        ; Continue until all extra bytes
      06 11 025D 604      BRB     90$            ; are checked
      FD9E' 30 025F 605 80$:  BSBW   FALS$UNS_KEY  ; Return error in Status message
      0324 31 0262 606      BRW     EXIT_FAILURE ; Exit state with failure
      0265 607 :
      0265 608 :
      0265 609 : Use the CTLFUNC field value as the next state table value.
      0265 610 :
      0265 611 :
      40 A9 90 0265 612 90$:  MOVB   DAP$B_CTLFUNC(R9),- ; Store new state transition value
      10 AB      0268 613      FALS$VALUE(R8)      ;
      031F 31 026A 614      BRW     EXIT_SUCCESS   ; Exit state with success
```

```

026D 616 .SBTTL FAL$DECODE_CON
026D 617
026D 618 :++
026D 619 : Process the Continue Transfer message which has been received and validated.
026D 620 :--
026D 621
026D 622 FAL$DECODE_CON:: ; Entry point
026D 623
026D 624 :
026D 625 : Use the CONFUNC field value as the next state transition table value.
026D 626 :
026D 627
40 A9 90 026D 628 MOVB DAP$B_CONFUNC(R9),- ; Store new state transition value
10 A8 0270 629 FAL$B_VALUE(R8) ;
0317 31 0272 630 BRW EXIT_SUCCESS ; Exit state with success

```



```
0275 632      .SBTTL  FAL$DECODE_CMP
0275 633
0275 634      :++
0275 635      : Process the Access Complete message which has been received and validated.
0275 636      : Update the FAB if necessary.
0275 637      :--
0275 638
0275 639 FAL$DECODE_CMP::      : Entry point
0275 640
0275 641      :
0275 642      : Process the DAP FOP field.
0275 643      : Do not update the FOP field in the FAB if this is a DAP DISCONNECT function
0275 644      : or if no FOP field was included in the Access Complete message.
0275 645      :
0275 646
51  44 A9  D0 0275 647      MOVL  DAP$L_FOP2(R9),R1      : Get DAP FOP bits
      03  13 0279 648      BEQL  10$      : Branch if no bits to map
      01C6 30 027B 649      BSBW  MAP_FOP_FIELD      : Update FOP in FAB
027E 650
027E 651      :
027E 652      : Use the CMPFUNC field value as the next state table value.
027E 653      :
027E 654
      40 A9  90 027E 655 10$:  MOVB  DAP$B_CMPFUNC(R9),-      : Store new state transition value
      10 A8      0281 656      :
      0306 31 0283 657      BRW   EXIT_SUCCESS      : Exit state with success
```

```
0286 659      .SBTTL  FAL$DECODE_KEY
0286 660
0286 661      ;++
0286 662      ; Process the Key Definition message which has been received and validated.
0286 663      ; Update the KEYXAB (by key of reference) with information from this message.
0286 664      ;--
0286 665
0286 666 FAL$DECODE_KEY::                                ; Entry point
0286 667
0286 668      ;
0286 669      ; Initialize the appropriate Key Definition XAB (in the FAL work area) and
0286 670      ; process the DAP REF field.
0286 671      ;
0286 672
0286 673      MOVZBL  DAP$B_REF(R9),R6                        ; Get key of reference value
0286 674      BSBW   FAL$INIT_KEYXAB                      ; On return R7 = address of XAB
0286 675      BLBS   R0,10$                                ; Branch on success
0286 676      BRW    EXIT_FAILURE                          ; Exit state with failure
0286 677      ;                                           ; (i.e., ignore this DAP message)
0286 678
0286 679      ;
0286 680      ; Process the DAP KNM field.
0286 681      ;
0286 682
0286 683      ASSUME   FAL$K_KEYNAM EQ 32
0286 684
0286 685 10$:      TSTL   DAP$Q_KNM(R9)                      ; Branch if no key name string was
0286 686          BEQL   20$                                ; specified
0286 687          MOVCS  DAP$Q_KNM(R9),-                      ; Copy DAP key name string
0286 688          @DAP$Q_KNM+4(R9),-                          ; to 32 byte key name buffer
0286 689          #0,#32,@XAB$B_KNM(R7)                      ; with zero fill
0286 690          BRB    30$
0286 691 20$:      CLRL   XAB$B_KNM(R7)                      ; Zero key name buffer address
0286 692
0286 693      ;
0286 694      ; Process the DAP FLG field.
0286 695      ;
0286 696
0286 697 30$:      MOVZBL  DAP$B_FLG(R9),R1                    ; Get DAP FLG bits
0286 698          CLRL   R2                                      ; Clear RMS FLG bits
0286 699          $MAPBIT  DAP$V_DUP,XAB$V_DUP                  ; Map DUP bit
0286 700          $MAPBIT  DAP$V_CHG,XAB$V_CHG                  ; Map CHG bit
0286 701          $MAPBIT  DAP$V_NUL_CHR,XAB$V_NUL              ; Map NUL bit
0286 702          MOVB   R2,XAB$B_FLG(R7)                      ; Update FLG field in XAB
0286 703
0286 704      ;
0286 705      ; Process the DAP DFL, IFL, NUL, IAN, LAN, DAN, and DTP fields.
0286 706      ;
0286 707
0286 708          MOVB   DAP$W_DFL(R9),XAB$W_DFL(R7)
0286 709          MOVB   DAP$W_IFL(R9),XAB$W_IFL(R7)
0286 710          MOVB   DAP$B_NUL(R9),XAB$B_NUL(R7)
0286 711          MOVB   DAP$B_IAN(R9),XAB$B_IAN(R7)
0286 712          MOVB   DAP$B_LAN(R9),XAB$B_LAN(R7)
0286 713          MOVB   DAP$B_DAN(R9),XAB$B_DAN(R7)
0286 714          MOVB   DAP$B_DTP(R9),XAB$B_DTP(R7)
0286 715
```

56 6C A9 9A 0286 673  
FD73' 30 028A 674  
03 50 E8 028D 675  
02F6 31 0290 676  
0293 677  
0293 678  
0293 679  
0293 680  
0293 681  
0293 682  
0293 683  
0293 684  
64 A9 D5 0293 685 10\$:  
0B 13 0296 686  
64 A9 2C 0298 687  
68 B9 029B 688  
38 B7 20 00 029D 689  
03 11 02A1 690  
38 A7 D4 02A3 691 20\$:  
02A6 692  
02A6 693  
02A6 694  
02A6 695  
02A6 696  
51 48 A9 9A 02A6 697 30\$:  
52 D4 02AA 698  
02AC 699  
02B4 700  
02BC 701  
12 A7 52 90 02C4 702  
02C8 703  
02C8 704  
02C8 705  
02C8 706  
02C8 707  
1C A7 44 A9 90 02C8 708  
1A A7 46 A9 90 02CD 709  
15 A7 6D A9 90 02D2 710  
08 A7 6E A9 90 02D7 711  
09 A7 6F A9 90 02DC 712  
0A A7 70 A9 90 02E1 713  
13 A7 71 A9 90 02E6 714  
02EB 715



```

02EB 716 :
02EB 717 : Process the DAP NSG, POS, and SIZ fields.
02EB 718 :
02EB 719 : Note: FALS$DECODE_MSG guarantees that 0 < DAP$B_NSIG < 9.
02EB 720 :
02EB 721 :
56 49 A9 9A 02EB 722      MOVZBL DAP$B_NSIG(R9),R6      ; Get # key segments
14 A7 56 90 02EF 723      MOVB   R6,XAB$B_NSIG(R7)      ; Update NSG field in XAB
5C A9 56 28 02F3 724      MOVCL   R6,DAP$B_SIZ(R9),-      ; Copy 1 to 8 key size values
                    2E A7 02F7 725      XAB$B_SIZ(R7)      ; to XAB
56 56 01 78 02F9 726      ASHL    #1,R6,R6              ; Double byte count
4C A9 56 28 02FD 727      MOVCL   R6,DAP$W_POS(R9),-      ; Copy 1 to 8 key position values
                    1E A7 0301 728      XAB$W_POS(R7)      ; to XAB
0303 729 :
0303 730 :
0303 731 : Ignore the DAP RVB, DVB, DBS, IBS, LVL, TKS, and MRL fields as these are not
0303 732 : inputs to RMS.
0303 733 :
0303 734 : Finish paper work and exit.
0303 735 :
0303 736 :
0303 737      $SETBIT #FALS$V_KEYXAB,FALS$W_RECEIVED(R8)
0281 31 0308 738      BRW      EXIT_SUCCESS              ; Denote XAB to add to XAB chain
                    0308 739      ; Exit state with success
```

```
030B 741 .SBTTL FALS$DECODE_ALL
030B 742
030B 743 ;++
030B 744 ; Process the Allocation message which has been received and validated.
030B 745 ; Update the ALLXAB (by AID) with information from this message.
030B 746 ;--
030B 747
030B 748 FALS$DECODE_ALL:: ; Entry point
030B 749
030B 750 ;
030B 751 ; Initialize the appropriate Allocation XAB (in the FAL work area) and
030B 752 ; process the DAP AID field.
030B 753 ;
030B 754
56 50 A9 9A 030B 755 MOVZBL DAP$B_AID(R9),R6 ; Get area ID value
FCEE' 30 030F 756 BSBW FALS$INIT_ALLXAB ; On return R7 = address of XAB
03 50 E8 0312 757 BLBS R0,10$ ; Branch on success
0271 31 0315 758 BRW EXIT_FAILURE ; Exit state with failure
0318 759 ; (i.e., ignore this DAP message)
0318 760
0318 761 ;
0318 762 ; Process the DAP ALN field.
0318 763 ;
0318 764
0318 765 ASSUME DAP$K_ANY EQ 0
0318 766 ASSUME DAP$K_CYL EQ XAB$C_CYL
0318 767 ASSUME DAP$K_LBN EQ XAB$C_LBN
0318 768 ASSUME DAP$K_VBN EQ XAB$C_VBN
0318 769
09 A7 44 A9 90 0318 770 10$: MOVB DAP$B_ALN(R9),XAB$B_ALN(R7)
031D 771
031D 772 ;
031D 773 ; Process the DAP AOP field.
031D 774 ;
031D 775
51 45 A9 9A 031D 776 MOVZBL DAP$B_AOP(R9),R1 ; Get DAP AOP bits
52 D4 0321 777 CLRL R2 ; Clear RMS AOP bits
0323 778 $MAPBIT DAP$V_HRD,XAB$V_HRD ; Map HRD bit
032B 779 $MAPBIT DAP$V_CBT2,XAB$V_CBT ; Map CBT bit
0333 780 $MAPBIT DAP$V_CTG2,XAB$V_CTG ; Map CTG bit
033B 781 $MAPBIT DAP$V_ONC,XAB$V_ONC ; Map ONC bit
08 A7 52 90 0343 782 MOVB R2,XAB$B_AOP(R7) ; Update AOP field in XAB
0347 783
0347 784 ;
0347 785 ; Process the DAP VOL, LOC, ALQ, BKZ, and DEQ fields.
0347 786 ;
0347 787
0A A7 42 A9 B0 0347 788 MOVW DAP$W_VOL(R9),XAB$W_VOL(R7)
0C A7 48 A9 D0 034C 789 MOVL DAP$L_LOC(R9),XAB$L_LOC(R7)
10 A7 4C A9 D0 0351 790 MOVL DAP$L_ALQ2(R9),XAB$L_ALQ(R7)
16 A7 51 A9 90 0356 791 MOVB DAP$B_BKZ(R9),XAB$B_BKZ(R7)
14 A7 52 A9 B0 035B 792 MOVW DAP$W_DEQ2(R9),XAB$W_DEQ(R7)
0360 793
0360 794 ;
0360 795 ; Finish paper work and exit.
0360 796 ;
0360 797
```



		0360	798	\$SETBIT #FAL\$V_ALLXAB,FAL\$W_RECEIVED(R8)	
		0365	799		; Denote XAB to add to XAB chain
0224	31	0365	800	BRW EXIT_SUCCESS	; Exit state with success

FALA  
Psec

PSEC  
-----  
A  
\$ABS  
FALS

Phas  
-----  
Init  
Comm  
Pass  
Symb  
Pass  
Symb  
Psec  
Cros  
Asse

The  
1112  
Ther  
1042  
36 p

Macr  
-----  
\$25  
- \$25  
TOTAL  
2114  
Ther  
MACF

```
0368 802 .SBTTL FALS$DECODE_TIM
0368 803
0368 804 ;++
0368 805 ; Process the Date and Time message which has been received and validated.
0368 806 ; Initialize both the DATXAB and RDTXAB and update them with information from
0368 807 ; this message. Other action routines will determine which of the two XABs to
0368 808 ; to use (or both) depending on the function that will be performed.
0368 809 ;--
0368 810
0368 811 FALS$DECODE_TIM:: ; Entry point
0368 812
0368 813 ;
0368 814 ; Initialize and fill-in the Date and Time XAB.
0368 815 ;
0368 816
FC95' 30 0368 817 BSBW FALS$INIT_DATXAB ; On return R7 = address of XAB
48 A9 7D 0368 818 MOVQ DAPSQ_CDT(R9),- ; Copy creation date and time
14 A7 036E 819 XABSQ_CDT(R7),- ; binary value to XAB
50 A9 7D 0370 820 MOVQ DAPSQ_RDT(R9),- ; Copy revision date and time
0C A7 0373 821 XABSQ_RDT(R7),- ; binary value to XAB
58 A9 7D 0375 822 MOVQ DAPSQ_EDT(R9),- ; Copy expiration date and time
1C A7 0378 823 XABSQ_EDT(R7),- ; binary value to XAB
60 A9 7D 037A 824 MOVQ DAPSQ_BDT(R9),- ; Copy backup date and time
24 A7 037D 825 XABSQ_BDT(R7),- ; binary value to XAB
42 A9 B0 037F 826 MOVW DAP$W_RVN(R9),- ; Store revision number value in XAB
08 A7 0382 827 XABS$W_RVN(R7) ;
0384 828
0384 829 ;
0384 830 ; Initialize and fill-in the Revision Date and Time XAB.
0384 831 ;
0384 832
FC79' 30 0384 833 BSBW FALS$INIT_RDTXAB ; On return R7 = address of XAB
50 A9 7D 0387 834 MOVQ DAPSQ_RDT(R9),- ; Copy revision date and time
0C A7 038A 835 XABSQ_RDT(R7),- ; binary value to XAB
42 A9 B0 038C 836 MOVW DAP$W_RVN(R9),- ; Store revision number value in XAB
08 A7 038F 837 XABS$W_RVN(R7) ;
0391 838
0391 839 ;
0391 840 ; Finish paper work and exit.
0391 841 ;
0391 842 ;
72 A8 14 A8 0391 843 BISW2 #<<FALS$M_DATXAB>:- ; Denote XABs to add to XAB chain
0395 844 <FALS$M_RDTXAB>:- ;
0395 845 0>,FALS$W_RECEIVED(R8) ;
01F4 31 0395 846 BRW EXIT_SUCCESS ; Exit state with success
```



```
0398 848 .SBTTL FALS$DECODE_PRO
0398 849
0398 850 :++
0398 851 : Process the Protection message which has been received and validated.
0398 852 : Update the PROXAB with information from this message.
0398 853 :--
0398 854
0398 855 FALS$DECODE_PRO::
0398 856 BSBW FALS$INIT_PROXAB ; Entry point
0398 857 CLRL XAB$L_UIC(R7) ; On return R7 = address of XAB
0398 858 MOVW #-1,XAB$W_PRO(R7) ; Initialize UIC and protection mask
0398 859 ; fields to [0,0] and -1. These mean
0398 860 ; use process UIC and default process
0398 861 ; protection in effect, respectively
0398 862
0398 863 : Process the DAP OWNER field.
0398 864 :
0398 865
0398 866 MOVQ DAP$Q_OWNER(R9),R4 ; Get descriptor of ASCII string
0398 867 CMPB (R5),#^A\ ; Branch if string does not begin
0398 868 BNEQ 30$ ; with bracket
0398 869 CMPB -1(R5)[R4],#^A\] \ ; Branch if string does not end
0398 870 BNEQ 30$ ; with bracket
0398 871 SUBL2 #2,R4 ; Discard brackets
0398 872 INCL R5
0398 873 LOCC #^A\,\,R4,(R5) ; Locate group-member delimiter
0398 874 BEQL 30$ ; Branch on failure
0398 875 SUBL3 R5,R1,R4 ; <R4,R5> => group string
0398 876 DECL R0 ; <R0,R1> => member string
0398 877 INCL R1
0398 878 CLRL -(SP) ; Allocate space from stack
0398 879 PUSHL SP ; Address of result
0398 880 PUSHL R1 ; Address of input string
0398 881 PUSHL R0 ; Size of input string
0398 882 CALLS #3,G^LIB$CVT_OTB ; Convert octal string to binary
0398 883 BLBC R0,20$ ; Branch on failure
0398 884 MOVW (SP),XAB$W_MBM(R7) ; Update member UIC value in XAB
0398 885 PUSHL SP ; Address of result
0398 886 PUSHL R5 ; Address of input string
0398 887 PUSHL R4 ; Size of input string
0398 888 CALLS #3,G^LIB$CVT_OTB ; Convert octal string to binary
0398 889 BLBC R0,10$ ; Branch on failure
0398 890 MOVW (SP),XAB$W_GRP(R7) ; Update group UIC value in XAB
0398 891 BRB 20$ ; UIC has been successfully converted
0398 892 10$: CLRW XAB$W_MBM(R7) ; GRP is invalid, so also discard MBM
0398 893 20$: CLRL (SP)+ ; Deallocate space from stack
0398 894
0398 895
0398 896 : Process the DAP PROSYS, PROOWN, PROGRP, PROWLD fields.
0398 897 :
0398 898
0398 899 30$: BITW #<<DAP$M_PROSYS>|- ; Use default file protection in effect
0398 900 <DAP$M_PROOWN>|- ; for the user process if all four
0398 901 <DAP$M_PROGRP>|- ; protection fields of the DAP
0398 902 <DAP$M_PROWLD>|- ; Protection message were defaulted
0398 903 0>,DAP$W_PROMENU(R9) ; (i.e., omitted from message)
0398 904 BEQL 40$ ; Branch if no fields explicitly sent
0398 905
```

FC65' 30  
OC A7 D4  
08 A7 FFFF 8F B0

54 48 A9 7D  
5B 8F 65 91  
4C 12  
5D 8F FF A544 91  
44 12  
54 02 C2  
55 D6  
65 54 2C 3A  
39 13  
54 51 55 C3  
50 D7  
51 D6  
7E D4  
5E DD  
51 DD  
50 DD  
00000000'GF 03 FB  
1D 50 E9  
OC A7 6E B0  
5E DD  
55 DD  
54 DD  
00000000'GF 03 FB  
06 50 E9  
OE A7 6E B0  
03 11  
OC A7 B4  
8E D4

03A4 866  
03A8 867  
03AC 868  
03AE 869  
03B4 870  
03B6 871  
03B9 872  
03BB 873  
03BF 874  
03C1 875  
03C5 876  
03C7 877  
03C9 878  
03CB 879  
03CD 880  
03CF 881  
03D1 882  
03D8 883  
03DB 884  
03DF 885  
03E1 886  
03E3 887  
03E5 888  
03EC 889  
03EF 890  
03F3 891  
03F5 892  
03F8 893  
03FA 894  
03FA 895  
03FA 896  
03FA 897  
03FA 898  
03FA 899  
03FE 900  
03FE 901  
03FE 902  
03FE 903  
03FE 904

MOVQ DAP\$Q\_OWNER(R9),R4  
CMPB (R5),#^A\[\]  
BNEQ 30\$  
CMPB -1(R5)[R4],#^A\]\]  
BNEQ 30\$  
SUBL2 #2,R4  
INCL R5  
LOCC #^A\,\,R4,(R5)  
BEQL 30\$  
SUBL3 R5,R1,R4  
DECL R0  
INCL R1  
CLRL -(SP)  
PUSHL SP  
PUSHL R1  
PUSHL R0  
CALLS #3,G^LIB\$CVT\_OTB  
BLBC R0,20\$  
MOVW (SP),XAB\$W\_MBM(R7)  
PUSHL SP  
PUSHL R5  
PUSHL R4  
CALLS #3,G^LIB\$CVT\_OTB  
BLBC R0,10\$  
MOVW (SP),XAB\$W\_GRP(R7)  
BRB 20\$  
CLRW XAB\$W\_MBM(R7)  
CLRL (SP)+  
BITW #<<DAP\$M\_PROSYS>|-  
<DAP\$M\_PROOWN>|-  
<DAP\$M\_PROGRP>|-  
<DAP\$M\_PROWLD>|-  
0>,DAP\$W\_PROMENU(R9)  
BEQL 40\$

```
50 04 00 50 A9 F0 0400 905      INSV DAP$W_PROSYS(R9),#0,#4,R0 ; Map system bits
50 04 04 52 A9 F0 0406 906      INSV DAP$W_PROOWN(R9),#4,#4,R0 ; Map owner bits
50 04 08 54 A9 F0 040C 907      INSV DAP$W_PROGRP(R9),#8,#4,R0 ; Map group bits
50 04 0C 56 A9 F0 0412 908      INSV DAP$W_PROWLD(R9),#12,#4,R0 ; Map world bits
08 A7 50 B0 0418 909      MOVW R0,XAB$W_PRO(R7) ; Update protection mask in XAB
      041C 910
      041C 911 ;
      041C 912 ; Finish paper work and exit.
      041C 913 ;
      041C 914
      041C 915 40$: $SETBIT #FALS$V_PROXAB,FALS$W_RECEIVED(R8)
0168 31 0421 916      ; Denote XAB to add to XAB chain
      0421 917      BRW EXIT_SUCCESS ; Exit state with success
```



```
0424 919 .SBTTL FALS$DECODE_NAM
0424 920
0424 921 ;++
0424 922 ; Process the name message which has been received and validated.
0424 923 ; Update FAB2 with information from this message.
0424 924 ;
0424 925 ; NOTE: At this time, only a rename operation will cause a Name message to be
0424 926 ; returned by FAL.
0424 927 ;--
0424 928
0424 929 FALS$DECODE_NAM::
5A 0800 C8 DE 0424 930 MOVAL FALS$_FAB2(R8),R10 ; Entry point
    44 A9 90 0429 931 MOV B DAPSQ_NAMESPEC(R9),- ; Put new filename FAB (FAB2) in R10
    34 AA 042C 932 FAB$_FNS(R10) ; Store size of new filespec string
    44 A9 28 042E 933 DAPSQ_NAMESPEC(R9),- ; in FAB2
    48 B9 0431 934 @DAPSQ_NAMESPEC+4(R9),- ; Copy the filespec string to buffer
    2C BA 0433 935 @FAB$_FNA(R10) ;
52 44 A9 7E 0435 936 MOVAQ DAPSQ_NAMESPEC(R9),R2 ; Get address of filename descriptor
    FBC4 30 0439 937 BSBW FALSLOG_REQNAM2 ; Log requested new name in print file
5A 0200 C8 DE 043C 938 MOVAL FALS$_FAB(R8),R10 ; Restore old filename FAB in R10
    0148 31 0441 939 BRW EXIT_SUCCESS ; Exit state with success
```

```
0444 941 .SBTTL SUPPORT ROUTINES
0444 942
0444 943
0444 944 .SBTTL MAP_FOP_FIELD
0444 945
0444 946 ;++
0444 947 ; This routine maps DAP FOP bits into RMS FOP bits and stores the result in
0444 948 ; the FOP field of the FAB.
0444 949 ;
0444 950 ; R1 contains the DAP bitmask on input.
0444 951 ; R2 is destroyed on output.
0444 952 ;--
0444 953
0444 954 MAP_FOP_FIELD:
52 D4 0444 955 CLRL R2 ; Entry point
51 D5 0446 956 TSTL R1 ; Clear RMS FOP bits
03 12 0448 957 BNEQ 10$ ; Examine FOP bitmask
0090 31 044A 958 BRW 20$ ; Begin mapping if any bits are set
044D 959 10$: $MAPBIT DAP$V_RWO,FAB$V_RWO ; Branch if there are no bits to map
0455 960 $MAPBIT DAP$V_RWC,FAB$V_RWC ; Map RWO bit
045D 961 $MAPBIT DAP$V_POS,FAB$V_POS ; Map RWC bit
0465 962 $MAPBIT DAP$V_CTG,FAB$V_CTG ; Map POS bit
046D 963 $MAPBIT DAP$V_SUP,FAB$V_SUP ; Map CTG bit
0475 964 $MAPBIT DAP$V_NEF,FAB$V_NEF ; Map SUP bit
047D 965 $MAPBIT DAP$V_TMP,FAB$V_TMP ; Map NEF bit
0485 966 $MAPBIT DAP$V_TMD,FAB$V_TMD ; Map TMP bit
048D 967 $MAPBIT DAP$V_DMO,FAB$V_DMO ; Map TMD bit
0495 968 $MAPBIT DAP$V_WCK,FAB$V_WCK ; Map DMO bit
049D 969 $MAPBIT DAP$V_RCK,FAB$V_RCK ; Map WCK bit
04A5 970 ; ***** $MAPBIT DAP$V_CIF,FAB$V_CIF ; Map RCK bit
04A5 971 $MAPBIT DAP$V_SQO,FAB$V_SQO ; Map CIF bit
04AD 972 $MAPBIT DAP$V_MXV,FAB$V_MXV ; Map SQO bit
04B5 973 $MAPBIT DAP$V_SPL,FAB$V_SPL ; Map MXV bit
04BD 974 $MAPBIT DAP$V_SCF,FAB$V_SCF ; Map SPL bit
04C5 975 $MAPBIT DAP$V_DLT,FAB$V_DLT ; Map SCF bit
04CD 976 $MAPBIT DAP$V_CBT,FAB$V_CBT ; Map DLT bit
04D5 977 ; ***** $MAPBIT DAP$V_DFW,FAB$V_DFW ; Map CBT bit
04D5 978 $MAPBIT DAP$V_TEF,FAB$V_TEF ; Map DFW bit
04DD 979 ; $MAPBIT DAP$V_OFP,FAB$V_OFP ; Map TEF bit
04DD 980 ; Note: this bit has no meaning here
04DD 981 ; because only primary filespec
04DD 982 ; is being given to RMS by FAL
04 04 AA 18 E1 04DD 983 20$: BBC #FAB$V_NAM,FAB$L_FOP(R10),30$ ; Preserve state of NAM bit in FOP
04 04 AA 52 D0 04E2 984 $SETBIT #FAB$V_NAM,R2 ; Update FOP field in FAB
05 04E6 985 30$: MOVL R2,FAB$L_FOP(R10) ; Exit
05 04EA 986 RSB
```



```
04EB 988 .SBTTL MAP_ROP_FIELD
04EB 989
04EB 990 ;+
04EB 991 ; This routine maps DAP ROP bits into RMS ROP bits and stores the result in
04EB 992 ; the ROP field of the RAB.
04EB 993 ;
04EB 994 ; R1 contains the DAP bitmask on input.
04EB 995 ; R2 is destroyed on output.
04EB 996 ; -
04EB 997
04EB 998 MAP_ROP_FIELD:
52 D4 04EB 999 CLRL R2 ; Entry point
51 D5 04ED 1000 TSTL R1 ; Clear RMS ROP bits
03 12 04EF 1001 BNEQ 10$ ; Examine ROP bitmask
0090 31 04F1 1002 BRW 20$ ; Begin mapping if any bits are set
04F4 1003 10$: $MAPBIT DAP$V_EOF,RAB$V_EOF ; Branch if there are no bits to map
04FC 1004 $MAPBIT DAP$V_FDL,RAB$V_FDL ; Map EOF bit
0504 1005 $MAPBIT DAP$V_UIF,RAB$V_UIF ; Map FDL bit
050C 1006 $MAPBIT DAP$V_LOA,RAB$V_LOA ; Map UIF bit
0514 1007 $MAPBIT DAP$V_ULK,RAB$V_ULK ; Map LOA bit
051C 1008 $MAPBIT DAP$V_TPT,RAB$V_TPT ; Map ULK bit
0524 1009 $MAPBIT DAP$V_RAH,RAB$V_RAH ; Map TPT bit
052C 1010 $MAPBIT DAP$V_WBH,RAB$V_WBH ; Map RAH bit
0534 1011 $MAPBIT DAP$V_KGE,RAB$V_KGE ; Map WBH bit
053C 1012 $MAPBIT DAP$V_KGT,RAB$V_KGT ; Map KGE bit
0544 1013 $MAPBIT DAP$V_NLK,RAB$V_NLK ; Map KGT bit
054C 1014 $MAPBIT DAP$V_RLK,RAB$V_RLK ; Map NLK bit
0554 1015 $MAPBIT DAP$V_ROPBIO,RAB$V_BIO ; Map RLK bit
055C 1016 $MAPBIT DAP$V_LIM,RAB$V_LIM ; Map BIO bit
0564 1017 $MAPBIT DAP$V_NXR,RAB$V_NXR ; Map LIM bit
056C 1018 $MAPBIT DAP$V_ROPWAT,RAB$V_WAT ; Map NXR bit
0574 1019 $MAPBIT DAP$V_RRL,RAB$V_RRL ; Map WAT bit
057C 1020 $MAPBIT DAP$V_REA,RAB$V_REA ; Map RRL bit
04 AB 52 D0 0584 1021 20$: MOVL R2,RAB$L_ROP(R1T) ; Map REA bit
05 0588 1022 RSB ; Update ROP field in RAB
; Exit
```

```
0589 1024      .SBTTL STATE EXIT ROUTINES
0589 1025
0589 1026 :++
0589 1027 : Exit state with failure.
0589 1028 :--
0589 1029
50  D4 0589 1030 EXIT_FAILURE:      ; Entry point
05  05 0589 1031      CLRL      R0   ; Signal state transition failure
058B 1032      RSB              ; Exit to state table manager
058C 1033
058C 1034 :++
058C 1035 : Exit state with success.
058C 1036 :--
058C 1037
50  01 058C 1038 EXIT_SUCCESS:      ; Entry point
05  05 058C 1039      MOVL      #1,R0 ; Signal state transition success
058F 1040      RSB              ; Exit to state table manager
0590 1041
0590 1042      .END              ; End of module
```



\$\$COUNT  
DAP\$B\_ACCFUNC  
DAP\$B\_ACCOPT  
DAP\$B\_AID  
DAP\$B\_ALN  
DAP\$B\_AOP  
DAP\$B\_BITCNT  
DAP\$B\_BKS  
DAP\$B\_BKZ  
DAP\$B\_BLKCNT  
DAP\$B\_BSZ  
DAP\$B\_CMPFUNC  
DAP\$B\_CONFUNC  
DAP\$B\_CTLFUNC  
DAP\$B\_DAN  
DAP\$B\_DATATYPE  
DAP\$B\_DBS  
DAP\$B\_DCODE\_FID  
DAP\$B\_DCODE\_MAC  
DAP\$B\_DCODE\_MSG  
DAP\$B\_DECVER  
DAP\$B\_DTP  
DAP\$B\_ECONUM  
DAP\$B\_FAC  
DAP\$B\_FILESYS  
DAP\$B\_FLAGS  
DAP\$B\_FLG  
DAP\$B\_FSZ  
DAP\$B\_IAN  
DAP\$B\_IBS  
DAP\$B\_KRF  
DAP\$B\_LAN  
DAP\$B\_LEN256  
DAP\$B\_LENGTH  
DAP\$B\_LVL  
DAP\$B\_NAMETYPE  
DAP\$B\_NSQ  
DAP\$B\_NUL  
DAP\$B\_ORG  
DAP\$B\_OSTYPE  
DAP\$B\_RAC  
DAP\$B\_RAT  
DAP\$B\_REF  
DAP\$B\_RFM  
DAP\$B\_SHR  
DAP\$B\_SIZ  
DAP\$B\_SIZ\_TMP  
DAP\$B\_STREAMID  
DAP\$B\_TKS  
DAP\$B\_TYPE  
DAP\$B\_USRNUM  
DAP\$B\_USRVER  
DAP\$B\_VERNUM  
DAP\$B\_X\_FIELD  
DAP\$C\_BCN  
DAP\$K\_ANY  
DAP\$K\_BLK\_FILE

= 00000003  
00000040  
00000041  
00000050  
00000044  
00000045  
00000035  
00000050  
00000051  
00000056  
00000052  
00000040  
00000040  
00000040  
00000070  
00000044  
0000007C  
00000019  
0000001B  
0000001A  
00000047  
00000071  
00000045  
00000042  
00000043  
00000031  
00000048  
00000051  
0000006E  
0000007D  
00000047  
0000006F  
00000034  
00000033  
0000007E  
00000040  
00000049  
0000006D  
00000045  
00000042  
00000046  
00000047  
0000006C  
00000046  
00000043  
0000005C  
0000004A  
00000032  
0000007F  
00000030  
00000046  
00000048  
00000044  
00000024  
000000C0  
= 00000000  
= 00000005

DAP\$K\_BLK\_VBN  
DAP\$K\_BLN  
DAP\$K\_CNF\_MSG  
DAP\$K\_CYL  
DAP\$K\_DECVER\_V  
DAP\$K\_ECONUM\_V  
DAP\$K\_FIX  
DAP\$K\_IDX  
DAP\$K\_KEY\_ACC  
DAP\$K\_LBN  
DAP\$K\_LOAD  
DAP\$K\_REL  
DAP\$K\_RFA\_ACC  
DAP\$K\_RMS32  
DAP\$K\_SEQ  
DAP\$K\_SEQ\_ACC  
DAP\$K\_SEQ\_FILE  
DAP\$K\_STG  
DAP\$K\_STM  
DAP\$K\_STMCR  
DAP\$K\_STMLF  
DAP\$K\_SYSCAP1\_V  
DAP\$K\_SYSCAP2\_V  
DAP\$K\_UDF  
DAP\$K\_USRNUM\_V  
DAP\$K\_USRVER\_V  
DAP\$K\_VAR  
DAP\$K\_VAXVMS  
DAP\$K\_VBN  
DAP\$K\_VERNUM\_V  
DAP\$K\_VFC  
DAP\$L\_ALQ1  
DAP\$L\_ALQ2  
DAP\$L\_ATTMENU  
DAP\$L\_CMWA  
DAP\$L\_CRC\_RSLT  
DAP\$L\_DCODE\_STS  
DAP\$L\_DEV  
DAP\$L\_DVB  
DAP\$L\_EBK  
DAP\$L\_FOP1  
DAP\$L\_FOP2  
DAP\$L\_HBK  
DAP\$L\_KEYMENU  
DAP\$L\_LOC  
DAP\$L\_MRN  
DAP\$L\_MSG\_MASK  
DAP\$L\_ROP  
DAP\$L\_RVB  
DAP\$L\_SBN  
DAP\$L\_SSPWA  
DAP\$L\_SSP\_CAP  
DAP\$L\_SSP\_FLG  
DAP\$L\_TEMP  
DAP\$M\_BITCNT  
DAP\$M\_BLKCNT  
DAP\$M\_CMPFMT

= 00000004  
000000C0  
= 00000001  
= 00000001  
= 00000004  
= 00000000  
= 00000001  
= 00000020  
= 00000001  
= 00000002  
= 000000FF  
= 00000010  
= 00000002  
= 00000003  
= 00000000  
= 00000000  
= 00000003  
= 00000000  
= 00000004  
= 00000006  
= 00000005  
= EFF67DF7  
= 00001962  
= 00000000  
= 00000000  
= 00000000  
= 00000002  
= 00000007  
= 00000003  
= 00000007  
= 00000003  
0000004C  
0000004C  
00000040  
00000030  
00000020  
00000018  
00000068  
00000078  
00000078  
00000064  
00000044  
00000074  
00000040  
00000048  
00000058  
0000001C  
00000050  
00000074  
0000007C  
00000080  
00000088  
00000084  
00000090  
= 00000008  
= 00000040  
= 00000008



FALACTMSG  
Symbol table

- STATE TABLE ACTION ROUTINES

D 12

16-SEP-1984 01:36:46 VAX/VMS Macro V04-00  
5-SEP-1984 01:16:21 [FAL.SRC]FALACTMSG.MAR;1

Page 29  
(17)

DAPSM\_DFTSPEC = 00000010  
DAPSM\_DMO = 00002000  
DAPSM\_DSP\_3NAM = 00000200  
DAPSM\_EMBEDDED = 00000010  
DAPSM\_GET = 00000002  
DAPSM\_GO\_NOGO = 00000010  
DAPSM\_IMAGE = 00000002  
DAPSM\_LOADIM = 00000001  
DAPSM\_LSA = 00000040  
DAPSM\_MACY11 = 00000080  
DAPSM\_MSE = 00000010  
DAPSM\_PROGRP = 00000008  
DAPSM\_PROOWN = 00000004  
DAPSM\_PROSYS = 00000002  
DAPSM\_PROWLD = 00000010  
DAPSM\_SEGMENT = 00000040  
DAPSM\_TMP1\$ = 00000020  
DAPSM\_TMP2\$ = 000000C0  
DAPSM\_TMP3\$ = 00020000  
DAPSM\_TMP4\$ = 01000000  
DAPSM\_TMP5\$ = F0000000  
DAPSM\_ZERO = 00000080  
DAPSQ\_ADT = 00000070  
DAPSQ\_BDT = 00000060  
DAPSQ\_CDT = 00000048  
DAPSQ\_DCODE\_FLG = 00000000  
DAPSQ\_EDT = 00000058  
DAPSQ\_FILESPEC = 00000044  
DAPSQ\_KEY = 00000048  
DAPSQ\_KNM = 00000064  
DAPSQ\_MSG\_BUF1 = 00000008  
DAPSQ\_MSG\_BUF2 = 00000010  
DAPSQ\_NAMESPEC = 00000044  
DAPSQ\_OWNER = 00000048  
DAPSQ\_PASSWORD = 00000050  
DAPSQ\_PDT = 00000068  
DAPSQ\_RDT = 00000050  
DAPSQ\_RUNSYS = 0000005C  
DAPSQ\_SYSCAP = 00000028  
DAPSQ\_SYSPEC = 00000038  
DAPSV\_APP = 00000007  
DAPSV\_BIGBLK = 00000014  
DAPSV\_BIO = 00000005  
DAPSV\_BLK = 00000003  
DAPSV\_BRO = 00000006  
DAPSV\_CBT = 00000017  
DAPSV\_CBT2 = 00000002  
DAPSV\_CHG = 00000001  
DAPSV\_CR = 00000001  
DAPSV\_CTG = 00000007  
DAPSV\_CTG2 = 00000001  
DAPSV\_DAPCRC = 00000015  
DAPSV\_DEL = 00000002  
DAPSV\_DLT = 00000016  
DAPSV\_DMO = 0000000D  
DAPSV\_DUP = 00000000  
DAPSV\_EOF = 00000000

DAPSV\_FDL = 00000001  
DAPSV\_FTN = 00000000  
DAPSV\_GET = 00000001  
DAPSV\_HRD = 00000000  
DAPSV\_KEY = 00000001  
DAPSV\_KGE = 00000009  
DAPSV\_KGT = 0000000A  
DAPSV\_KRF = 00000002  
DAPSV\_LIM = 0000000E  
DAPSV\_LOA = 00000004  
DAPSV\_LOAD = 00000000  
DAPSV\_MSGBLK = 00000012  
DAPSV\_MXV = 00000013  
DAPSV\_NEF = 00000009  
DAPSV\_NIL = 00000006  
DAPSV\_NLK = 0000000B  
DAPSV\_NUL\_CHR = 00000002  
DAPSV\_NXR = 0000000F  
DAPSV\_ONC = 00000003  
DAPSV\_POS = 00000003  
DAPSV\_PRN = 00000002  
DAPSV\_PUT = 00000000  
DAPSV\_RAC = 00000000  
DAPSV\_RAH = 00000007  
DAPSV\_RCK = 0000000F  
DAPSV\_REA = 00000012  
DAPSV\_RLK = 0000000C  
DAPSV\_ROP = 00000003  
DAPSV\_ROPBIO = 0000000D  
DAPSV\_ROPWAT = 00000010  
DAPSV\_RRL = 00000011  
DAPSV\_RWC = 00000001  
DAPSV\_RWO = 00000000  
DAPSV\_SCF = 00000015  
DAPSV\_SHRDEL = 00000002  
DAPSV\_SHRGET = 00000001  
DAPSV\_SHRPUT = 00000000  
DAPSV\_SHRUPD = 00000003  
DAPSV\_SPL = 00000014  
DAPSV\_SQO = 00000012  
DAPSV\_SUP = 00000008  
DAPSV\_TEF = 0000001A  
DAPSV\_TMD = 0000000B  
DAPSV\_TMP = 0000000A  
DAPSV\_TPT = 00000006  
DAPSV\_TRN = 00000004  
DAPSV\_UIF = 00000002  
DAPSV\_ULK = 00000005  
DAPSV\_UPD = 00000003  
DAPSV\_UPI = 00000005  
DAPSV\_VAXVMS = 00000034  
DAPSV\_WBH = 00000008  
DAPSV\_WCK = 0000000E  
DAPSW\_ALLMENU = 00000040  
DAPSW\_BLS = 00000048  
DAPSW\_BUFSIZ = 00000040  
DAPSW\_CHECK = 00000042

FAL  
V04.



FALACTMSG  
Symbol table

- STATE TABLE ACTION ROUTINES

E 12

16-SEP-1984 01:36:46 VAX/VMS Macro V04-00  
5-SEP-1984 01:16:21 [FAL.SRC]FALACTMSG.MAR;1

Page 30  
(17)

DAPSW_CTLMENU	00000044			FABSV_DMO	= 0000000C		
DAPSW_DEQ1	00000054			FABSV_EXE	= 00000007		
DAPSW_DEQ2	00000052			FABSV_FTN	= 00000000		
DAPSW_DFL	00000044			FABSV_GET	= 00000001		
DAPSW_DISPLAY1	0000004C			FABSV_MXV	= 00000001		
DAPSW_DISPLAY2	00000054			FABSV_NAM	= 00000018		
DAPSW_FFB	00000072			FABSV_NEF	= 0000000A		
DAPSW_IFL	00000046			FABSV_NIL	= 00000005		
DAPSW_LRL	00000070			FABSV_POS	= 00000008		
DAPSW_MRL	00000072			FABSV_PRN	= 00000002		
DAPSW_MRS	0000004A			FABSV_PUT	= 00000000		
DAPSW_PARTNER	00000006			FABSV_RCK	= 00000017		
DAPSW_POS	0000004C			FABSV_RWC	= 0000000B		
DAPSW_POS_TMP	0000004A			FABSV_RWO	= 00000007		
DAPSW_PROGRP	00000054			FABSV_SCF	= 0000000E		
DAPSW_PROMENU	00000040			FABSV_SHRDEL	= 00000002		
DAPSW_PROOWN	00000052			FABSV_SHRGET	= 00000001		
DAPSW_PROSYS	00000050			FABSV_SHRPUT	= 00000000		
DAPSW_PROWLD	00000056			FABSV_SHRUPD	= 00000003		
DAPSW_RVN	00000042			FABSV_SPL	= 0000000D		
DAPSW_SSP_MENU	00000080			FABSV_SQO	= 00000006		
DAPSW_TIMENU	00000040			FABSV_SUP	= 00000002		
DAPSW_VERSION	00000004			FABSV_TEF	= 0000001C		
DAPSW_VOL	00000042			FABSV_TMD	= 00000004		
EXIT_FAILURE	00000589	R	02	FABSV_TMP	= 00000003		
EXIT_SUCCESS	0000058C	R	02	FABSV_TRN	= 00000004		
FABSB_BKS	= 0000003E			FABSV_UPD	= 00000003		
FABSB_FAC	= 00000016			FABSV_UPI	= 00000006		
FABSB_FNS	= 00000034			FABSV_WCK	= 00000009		
FABSB_FSZ	= 0000003F			FABSW_BLS	= 0000003C		
FABSB_URG	= 0000001D			FABSW_DEQ	= 00000014		
FABSB_RAT	= 0000001E			FABSW_MRS	= 00000036		
FABSB_RFM	= 0000001F			FALSBUILD_HEAD	*****	X	02
FABSB_SHR	= 00000017			FALSBUILD_TAIL	*****	X	02
FABSC_FIX	= 00000001			FALSB_ACCFUNC	000001F6		
FABSC_IDX	= 00000020			FALSB_ACCOPT	000001F5		
FABSC_REL	= 00000010			FALSB_DATATYPE	000001F4		
FABSC_SEQ	= 00000000			FALSB_DISABLE	00000006		
FABSC_STM	= 00000004			FALSB_ENABLE	00000005		
FABSC_STMCR	= 00000006			FALSB_LOGGING	00000004		
FABSC_STMLF	= 00000005			FALSB_MISCOPT	00000007		
FABSC_UDF	= 00000000			FALSB_RAC	000001F7		
FABSC_VAR	= 00000002			FALSB_RBK_CACHE	00000012		
FABSC_VFC	= 00000003			FALSB_RCVBUFIDX	00000011		
FABSL_ALQ	= 00000010			FALSB_VALUE	00000010		
FABSL_FNA	= 0000002C			FALSCVT_BN8_EXT	*****	X	02
FABSL_FOP	= 00000004			FALSC_WRKBLN	00002000		
FABSL_MRN	= 00000038			FALSDCODE_ACC	0000010B	RG	02
FABSM_CR	= 00000002			FALSDCODE_ALL	0000030B	RG	02
FABSV_BIO	= 00000005			FALSDCODE_ATT	000000A8	RG	02
FABSV_BLK	= 00000003			FALSDCODE_CMP	00000275	RG	02
FABSV_BRO	= 00000006			FALSDCODE_CNF	00000000	RG	02
FABSV_CBT	= 00000015			FALSDCODE_CON	0000026D	RG	02
FABSV_CR	= 00000001			FALSDCODE_CTL	00000196	RG	02
FABSV_CTG	= 00000014			FALSDCODE_KEY	00000286	RG	02
FABSV_DEL	= 00000002			FALSDCODE_NAM	00000424	RG	02
FABSV_DLT	= 0000000F			FALSDCODE_PRO	00000398	RG	02



Variable	Value	RG	02
FAL\$DECODE TIM	00000368		02
FAL\$INIT_ACLXAB	*****	X	02
FAL\$INIT_DATXAB	*****	X	02
FAL\$INIT_KEYXAB	*****	X	02
FAL\$INIT_PROXAB	*****	X	02
FAL\$INIT_RDTXAB	*****	X	02
FAL\$K_KEYNAM	= 00000020		
FAL\$K_WRKBLN	00002000		
FAL\$LOG_REQNAM	*****	X	02
FAL\$LOG_REQNAM2	*****	X	02
FAL\$SL_ACLXAB	00000C00		
FAL\$SL_ALLXABINI	00000074		
FAL\$SL_CHAIN NXT	0000007C		
FAL\$SL_DATXAB	00000320		
FAL\$SL_FAB	00000200		
FAL\$SL_FAB2	00000800		
FAL\$SL_FHCXAB	000002F4		
FAL\$SL_FOP	000001F8		
FAL\$SL_KEYNAM	00001C00		
FAL\$SL_KEYXAB	00001000		
FAL\$SL_KEYXABINI	00000078		
FAL\$SL_NAM	00000294		
FAL\$SL_NAM2	00000850		
FAL\$SL_NUMBER	000001FC		
FAL\$SL_PROXAB	0000034C		
FAL\$SL_RAB	00000250		
FAL\$SL_RCVBUF	0000005C		
FAL\$SL_RDTXAB	000003B0		
FAL\$SL_RMS_PTR	0000006C		
FAL\$SL_STB	000000C0		
FAL\$SL_SUMXAB	000003A4		
FAL\$SL_TEMP	000003F4		
FAL\$SL_USE_SC1	000000A8		
FAL\$SL_USE_SC2	000000AC		
FAL\$SL_USE_VER	000000A4		
FAL\$M_DATXAB	= 00000004		
FAL\$M_RDTXAB	= 00000010		
FAL\$Q_BLD	00000050		
FAL\$Q_DIRNAME	00000088		
FAL\$Q_FALLOG	00000090		
FAL\$Q_FLG	00000000		
FAL\$Q_MBX	00000038		
FAL\$Q_MBXIOSB	00000030		
FAL\$Q_RCV	00000040		
FAL\$Q_RCVIOSB	00000020		
FAL\$Q_RMS	00000064		
FAL\$Q_STATE_CTX	00000008		
FAL\$Q_SYSNET	00000098		
FAL\$Q_TEMP	000003F8		
FAL\$Q_VOLNAME	00000080		
FAL\$Q_XMT	00000048		
FAL\$Q_XMTIOSB	00000028		
FAL\$TRANSMIT	*****	X	02
FAL\$T_DAP	00000100		
FAL\$T_DIRNAME	00001F00		
FAL\$T_EXPAND	00000500		
FAL\$T_EXPAND2	00000A00		

FALST_FALLOG		00001C00		
FALST_FILESPEC		00000400		
FALST_FILESPEC2		00000900		
FALST_KEYBUF		00000700		
FALST_MBXBUF		00001980		
FALST_PRTBUF1		00001A00		
FALST_PRTBUF2		00001B00		
FALST_RESULT		00000600		
FALST_RESULT2		00000B00		
FALST_SYSNET		00001D00		
FALST_VOLNAME		00001E00		
FALSUNS_KEY		*****	X	02
FALSV_ACLXAB	=	00000001		
FALSV_ATT_MSG	=	00000001		
FALSV_BLK_IO	=	00000009		
FALSV_CNF_MSG	=	00000000		
FALSV_DIS_CRC	=	00000030		
FALSV_DIS_MBK	=	00000031		
FALSV_FTM	=	00000008		
FALSV_KEYXAB	=	00000000		
FALSV_LAST_MSG	=	00000018		
FALSV_PROXAB	=	00000003		
FALSV_USE_DBS	=	00000039		
FALSV_USE_SC1	=	0000003C		
FALSV_USE_SC2	=	0000003D		
FALSV_USE_SYS	=	0000003A		
FALSV_USE_VER	=	0000003B		
FALSV_WILD	=	0000000A		
FALSW_DAPBUFSIZ		0000001A		
FALSW_DISPLAY		00000070		
FALSW_LNKCHN		0000001C		
FALSW_MBXCHN		0000001E		
FALSW_QIOBUFSIZ		00000018		
FALSW_RECEIVED		00000072		
FALSW_USE_DBS		000000A0		
FALSW_USE_SYS		000000A2		
KEY_FIELD		000001FE	R	02
KRF_FIELD		000001F4	R	02
LIBSCVT_OTB		*****	X	02
MAP_FOP_FIELD		00000444	R	02
MAP_ROP_FIELD		000004EB	R	02
RAB\$B_KRF	=	00000035		
RAB\$B_KSZ	=	00000034		
RAB\$B_RAC	=	0000001E		
RAB\$C_KEY	=	00000001		
RAB\$C_RFA	=	00000002		
RAB\$C_SEQ	=	00000000		
RAB\$L_BKT	=	00000038		
RAB\$L_KBF	=	00000030		
RAB\$L_ROP	=	00000004		
RAB\$V_BIO	=	0000000B		
RAB\$V_EOF	=	00000008		
RAB\$V_FDL	=	00000006		
RAB\$V_KGE	=	00000015		
RAB\$V_KGT	=	00000016		
RAB\$V_LIM	=	0000000E		
RAB\$V_LOA	=	0000000D		

FAL  
Sym[illegible]



RABSV_NLK	=	00000014		
RABSV_NXR	=	00000017		
RABSV_RAH	=	00000009		
RABSV_REA	=	00000002		
RABSV_RLK	=	00000013		
RABSV_RRL	=	00000003		
RABSV_TPT	=	00000001		
RABSV_UIF	=	00000004		
RABSV_ULK	=	00000012		
RABSV_WAT	=	00000011		
RABSV_WBH	=	0000000A		
RABSW_RFA	=	00000010		
RAC_FIELD		0000019F	R	02
ROP_FIELD		000001E8	R	02
SEND_CNF		00000015	R	02
XABSB_ALN	=	00000009		
XABSB_AOP	=	00000008		
XABSB_BKZ	=	00000016		
XABSB_DAN	=	0000000A		
XABSB_DTP	=	00000013		
XABSB_FLG	=	00000012		
XABSB_IAN	=	00000008		
XABSB_LAN	=	00000009		
XABSB_NSG	=	00000014		
XABSB_NUL	=	00000015		
XABSB_SIZ	=	0000002E		
XABSC_CYL	=	00000001		
XABSC_LBN	=	00000002		
XABSC_VBN	=	00000003		
XABSL_ALQ	=	00000010		
XABSL_KNM	=	00000038		
XABSL_LOC	=	0000000C		
XABSL_UIC	=	0000000C		
XABSQ_BDT	=	00000024		
XABSQ_CDT	=	00000014		
XABSQ_EDT	=	0000001C		
XABSQ_RDT	=	0000000C		
XABSV_CBT	=	00000005		
XABSV_CHG	=	00000001		
XABSV_CTG	=	00000007		
XABSV_DUP	=	00000000		
XABSV_HRD	=	00000000		
XABSV_NUL	=	00000002		
XABSV_ONC	=	00000001		
XABSW_DEQ	=	00000014		
XABSW_DFL	=	0000001C		
XABSW_GRP	=	0000000E		
XABSW_IFL	=	0000001A		
XABSW_LRL		*****	X	02
XABSW_MBM	=	0000000C		
XABSW_POS	=	0000001E		
XABSW_PRO	=	00000008		
XABSW_RVN	=	00000008		
XABSW_VOL	=	0000000A		

[illegible]

+-----+  
! Psect synopsis !  
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
\$AB\$\$	00002000 ( 8192.)	01 ( 1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE				
FAL\$CODE	00000590 ( 1424.)	02 ( 2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE				

+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.05	00:00:01.61
Command processing	142	00:00:00.36	00:00:02.39
Pass 1	450	00:00:13.74	00:00:51.94
Symbol table sort	0	00:00:01.65	00:00:04.98
Pass 2	192	00:00:02.86	00:00:10.29
Symbol table output	59	00:00:00.30	00:00:00.60
Psect synopsis output	1	00:00:00.03	00:00:00.73
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	883	00:00:19.00	00:01:12.55

The working set limit was 1950 pages.  
111221 bytes (218 pages) of virtual memory were used to buffer the intermediate code.  
There were 90 pages of symbol table space allocated to hold 1637 non-local and 110 local symbols.  
1042 source lines were read in Pass 1, producing 17 object records in Pass 2.  
36 pages of virtual memory were used to define 35 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
\$255\$DUA28:[FAL.OBJ]FAL.MLB;1	20
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	32

2114 GETS were required to define 32 macros.  
There were no errors, warnings or information messages.  
MACRO/LIS=LISS:FALACTMSG/OBJ=OBJ\$:FALACTMSG MSRC\$:FALACTMSG/UPDATE=(ENH\$:FALACTMSG)+LIB\$:FAL/LIB



0174 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

